#20
KWS
11-16-01
NE

RESPONSE UNDER 37 CFR 1.116
EXPEDITED PROCEDURE
EXAMINING GROUP 2132
Docket No.: 826.1377

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Hironobu KITAJIMA et al.

Serial No. 08/814,409 Group Art Unit: 2132

Confirmation No. 4623

Filed: March 11, 1997 Examiner: D. Meislahn

For: ENCRYPTING/DECRYPTING SYSTEM WITH PROGRAMMABLE LOGIC DEVICE/UNIT AND METHOD THEREOF

REQUEST FOR RECONSIDERATION

Assistant Commissioner for Patents
Washington, D.C. 20231

Attention: **BOX AF**

Do Not Enter 16 Nov 2001 KW

Sir:

This is in response to the Final Office Action mailed August 13, 2001, and having a period for response set to expire on November 13, 2001. In the Office Action, the Examiner noted that claims 1-4, 6-13 and 15-31 are pending in the application; rejected claims 1-4, 6-8, 10-13, 15-17 and 19-22 under 35 U.S.C. § 103(a); and rejected claims 23-31 under 35 U.S.C. § 102(b). In rejecting the claims, U.S. Patents 4,972,478 to Dabbish; 5,345,508 to Lynn et al. (References G and E in the March 23, 1999 Office Action, respectively); and 5,499,192 to Knapp et al. (Reference A in the January 26, 2001 Office Action) and the Third Edition of the Microsoft Press Computer Dictionary (Exhibit A attached to the May 29, 2001 Amendment) were cited. The Examiner's rejections are traversed below.

Teachings of Cited References

In item 5 on pages 3-4 of the Office Action, claims 1-4, 6, 8, 10-13, 15, 17 and 19-22 were rejected under 35 U.S.C. § 103(a) as unpatentable over Dabbish '478 in view of Knapp et al. and the Microsoft Press Computer Dictionary, 3rd ed. (hereafter MS Computer Dictionary). The cited portions of Dabbish '478 were the Abstract; column 1, lines 51-67; and column 3, lines 44-46. The Abstract and columns 1-4 of Knapp et al. were cited as teaching "mapping data to

programmable logic devices" and the MS Computer Dictionary was cited as teaching that object-oriented programming was old in the art. At page 4, lines 4-5, official notice was relied upon that it was known to shield electrical components using an enclosure and at page 4, lines 17-18, official notice was relied upon that "updating keys or other cryptographic devices is old and well known." However, nothing was cited as teaching or suggesting putting any specific components within an enclosure.

Applicants acknowledge that the cryptographic circuit taught by the Dabbish '478 can be programmed with various cipher algorithms (as discussed in the Abstract of Dabbish '478). However, one distinction between the present invention and the prior art is the technique used for programming the circuit taught by Dabbish '478 and what changes in the technique would be obvious to one of ordinary skill in the art from Knapp et al., whether using object-oriented principles or not. The cited portion of column 3 in Dabbish '478 does not programming techniques since it only mentions that decryption is performed in a manner similar to encryption. The cited portion in column 1 of Dabbish '478 is in the Summary of the Invention section and thus, it provides only an overview of how the circuit taught by Dabbish '478 works. To understand what the terms mean in column 1 of Dabbish '478, a person of ordinary skill in the art would have to read the detailed description provided in columns 2-4. Following is a summary of the seven step procedure described at column 2, line 48 to column 3, line 24 in the "Best Mode for Carrying Out the Invention" section of Dabbish '478.

According to Dabbish '478, the RAM 188 in cryptographic circuit 10 "receives preliminary storage instructions from the external programming equipment (EPE) (105) and stores them per the initial instructions" (column 2, lines 48-50) which are previously stored in bootstrap loader 119. The preliminary storage instructions control storage of "cipher algorithm storage instructions" (column 2, lines 56-57) which are stored in EEPROM 117. "The cryptographic cores (100 & 101) receive a cipher algorithm from the EPE (105) and store it per the cipher algorithm storage instructions" (column 2, lines 59-61). The EPE 105 then performs a storage test and sends "preliminary execution instructions" to RAM 118 in step 5 (see column 3, lines 3-10). The EEPROM 117 then stores "cipher algorithm executions (sic) instructions" received from the EPE 105, according to the preliminary execution instructions (see column 3, lines 11-18). Finally, in step 7, the EPE 105 performs an execution test and the cryptographic circuit 10 is ready to encrypt (and decrypt) security sensitive digital information.

It is clear from the detailed description of Dabbish '478 that the EPE 105 provides low-level instructions to program the cryptographic circuit 10. As discussed above, the EPE 105

provides preliminary storage instructions that are stored in RAM, then cipher algorithm storage instructions that are stored in accordance with the preliminary storage instructions, and finally a cipher algorithm that is stored in accordance with the cipher algorithm storage instructions. The only instructions required to change the cipher algorithm that are not provided by the EPE 105 are the "initial instructions ... stored in the bootstrap loader (119)" (column 2, lines 51-52).

In item 4 on page 2 of the Office Action in the Response to Arguments, it is stated that the "EPE anticipates a remote computer, (and) the I/O circuit reads on a network connecting unit" (Office Action, page 2, lines 13-14). It This statement indicates that the EPE 105 is being read on the "remote computer" recited in the preamble of claim 1. This is consistent with the statement in the Office Action that column 1, lines 51-67 of Dabbish '478 teaches "that orders to change the encryption algorithm originate from sources external to the apparatus" (Office Action, page 3, lines 19-20). Claim 1 was amended on October 17, 2001 by the addition of "an enclosure", to clarify what is "external to the apparatus". As now recited in claim 1, the circuit unit, network connecting unit, mapping data generating unit and changing unit are all inside the enclosure (see claim 1, last two lines). The Examiner apparently has acknowledged that the EPE 105, as indicated by its full name, "external programming equipment" (Dabbish '478, column 2, line 20) is not inside any enclosure that might exist in the device taught by Dabbish '478.

Rejections under 35 U.S.C. § 103(a)

The rejection set forth on pages 3-4 of the Office Action does not clearly set forth what components in Dabbish '478 correspond to the elements recited in claim 1. The only correspondence between components in Dabbish '478 and the elements in claim 1 that has been found is provided in item 4 on page 2 of the Office Action where it is indicated that the EPE 105 corresponds to the remote computer and the I/O circuit 103 corresponds to the network connecting unit. It is submitted that this correspondence breaks down, because the EPE 105 performs operations similar to the mapping data generating unit. As noted above in the description of the seven step procedure disclosed by Dabbish '478, the EPE 105 provides detailed instructions on how to encrypt data, not merely "predetermined criteria received from the remote computer" (claim 1, lines 9-10) used by the mapping data generating unit "to generate a mapping data object representing the structure of the encrypting circuit" (claim 1, lines 10-11). Once it is acknowledged that the EPE 105 would be outside any enclosure surrounding the cryptographic circuit taught by Dabbish '478, there is nothing left in the cryptographic circuit taught by Dabbish '478 to perform the operations of the mapping data generating unit recited in claim 1.

In the Office Action, it was asserted that the functions performed by the mapping data generating unit would be obvious from what is taught by Knapp et al. and MS Computer Dictionary. However, there is no indication of what in the device taught by Dabbish '478 could be modified using the teachings of Knapp et al. to perform the operations of the mapping data generating unit recited in claim 1. Claim 1 requires that something inside an enclosure reads "change data for changing at least one of the encrypting specifications in accordance with predetermined criteria received from" (claim 1, lines 8-9) outside the encrypting apparatus and generates "a mapping data object representing the structure of the encrypting circuit" (claim 1, lines 10-11). Nothing has been cited to suggest why one of ordinary skill in the art would modify the internal components of the circuit taught by Dabbish '478 to include reading change data and generating an object representing the structure of the encrypting circuit.

If it is the position of the Examiner that it would be obvious to one of ordinary skill in the art to move the functions of EPE 105 inside an enclosure, there are a number of questions that would need to be answered. For example, where is the suggestion in prior art that the EPE 105 receives "predetermined criteria ... from ... (a) remote computer" (claim 1, lines 9-10)? Furthermore, how is it possible to avoid destroying the entire purpose of the circuit taught by Dabbish '478 of a secure encryption/decryption device if the components used for programming the device are included inside the enclosure of the circuit? Nothing has been cited by the Examiner or found in any of the references suggesting that a change instruction is received via communication circuitry 104 to provide the predetermined criteria recited in claim 1 for use by EPE 105 in performing the operations disclosed in Dabbish '478 to change the encryption algorithm. For the above reasons, it is submitted that claim 1 patentably distinguishes over Dabbish '478 in view of Knapp et al. and MS Computer Dictionary.

Claims 2-4, 6 and 8 depend from claim 1 and therefore, it is submitted that claims 2-4, 6 and 8 patentably distinguish over the prior art set forth above with respect to claim 1. Furthermore, for the reasons set forth above, it is unclear why Knapp et al. would suggest to one of ordinary skill in the art how to modify the circuit taught by Dabbish '478 to meet the limitations recited in claims 2-4, since all three claims add details regarding operations performed by the mapping data generating unit and it is unclear what internal component of the circuit taught by Dabbish '478 could be modified to perform the operations taught by Knapp et al. Therefore, it is submitted that claims 2-4 further patentably distinguish over the prior art due to the details recited therein.

In addition, nothing has been cited in the prior art suggesting that data either received or transmitted by EPE 105 is "encrypted change data" (claim 6, line 2). Since the EPE 105 and internal components of the cryptographic circuit 10 are under the control of the person programming the unit, there would be no need for encryption of data transmitted by EPE 105 to the internal components of circuit 10. Furthermore, there is no suggestion in Dabbish '478 of any change data being transmitted over communication circuitry 104 and therefore, there is no need to encrypt any change data. For the above reasons, it is submitted that claim 6 further patentably distinguishes over the prior art cited in rejecting the claims.

Like claim 1, claim 10 recites "an enclosure substantially surrounding said circuit unit, said mapping data generating unit and said changing unit" (claim 10, last 2 lines). Similar limitations are recited in the last element of claims 19-22. For the reasons set forth above with respect to claim 1, it is submitted that claim 10, claims 19-22 and claims 11-13, 15 and 17 which depend from claim 10, patentably distinguish over the prior art used to reject the claims.

In item 6 on page 4 of the Office Action, claims 6 and 13 were rejected over the same combination of prior art used to reject claims 1 and 10, with the addition of official notice regarding "updating keys or other cryptographic devices" (page 4, lines 17-18). Nothing in item 6 suggests how an internal component of the circuit taught by Dabbish '478 could be modified to meet the limitations of the mapping data generating unit recited in claims 1 and 10. Since claims 7 and 16 depend from claims 1 and 10, respectively, it is submitted that claims 7 and 16 patentably distinguish over the prior art for the reasons set forth above with respect to claims 1 and 10.

In item 7 on page 5 of the Office Action, claims 9 and 18 were rejected under 35 U.S.C. § 103(a) as unpatentable over Dabbish '478, Knapp et al. and MS Computer Dictionary and further in view of Lynn et al. Nothing was cited in Lynn et al. suggesting the addition of or modification to an existing internal component taught by Dabbish '478 to perform the operations of the mapping data generating unit. Therefore, it is submitted that claims 9 and 18 patentably distinguish over the prior art used to reject these claims for the reasons set forth above with respect to claims 1 and 10 from which they depend.

Rejection under 35 U.S.C. § 102(b)

No explanation was provided in the August 13, 2001 Office Action regarding how claims 23-31 are anticipated by Dabbish '478. In particular, nowhere was it suggested that there is a component within the supervisory circuit or any other internal component that performs the function of "reading change data from a remote computer" (as recited on line 4 of claims 23 and

24.) In the common parlance of computer communications, the EPE 105 performs a "push" function, i.e. it supplies the preliminary storage instructions which are stored in RAM 118 in accordance with initial instructions stored in bootstrap loader 119. There is no suggestion of anything else in cryptographic circuit 10 reading data from the EPE 105. There is no suggestion that instructions should be stored in the bootstrap loader 119 to cause any internal component of the cryptographic circuit 10 to read data from EPE 105 or from some other remote device connected to communication circuitry 104. Since the changing operation recited in claims 23 and 24 relies on the change data read from the remote computer, no suggestion of the methods recited in claims 23 and 24 can be found in Dabbish '478.

Similarly, the final limitation recited in claims 25-31 each includes reading change data from a remote device. Therefore, it is submitted that claims 25-31 patentably distinguish over Dabbish '478 for the reasons discussed above with respect to claims 23 and 24.

SUMMARY

It is submitted that the references cited by the Examiner, taken alone or in combination and modified with any of the official notice taken by the Examiner do not teach or suggest the features of the present claimed invention. Thus, it is submitted that claims 1-4, 6-13 and 15-31 are in a condition suitable for allowance. Reconsideration of the claims and an early Notice of Allowance are earnestly solicited.

If there are any additional fees associated with this Request for Reconsideration, please charge the same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 11/13/01

By: Richard A. Gollhofer
Richard A. Gollhofer
Registration No. 31,106

700 Eleventh Street, NW, Suite 500
Washington, D.C. 20001
(202) 434-1500

Windows NT® Windows® 95

The Ultimate Computer Reference

The Comprehensive Standard for Business, School, Library, and Home

CD-ROM Included

Over 7,600 Terms

Additional Terms Available On Line Quarterly

Microsoft Press® Computer Dictionary

Third Edition

- **Over 300 illustrations and diagrams**
- **Extensive Internet coverage**
- **Featured in Microsoft® Bookshelf®**
- **Covers software, hardware, concepts, and more!**

Microsoft® Press

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Press Computer Dictionary. -- 3rd ed.
p. cm.
ISBN 1-57231-446-X
1. Computers--Dictionaries. 2. Microcomputers--Dictionaries.
I. Microsoft Press.
QA76.15.M54 1997
004'.03--dc21 97-15489
CIP

Printed and bound in the United States of America.

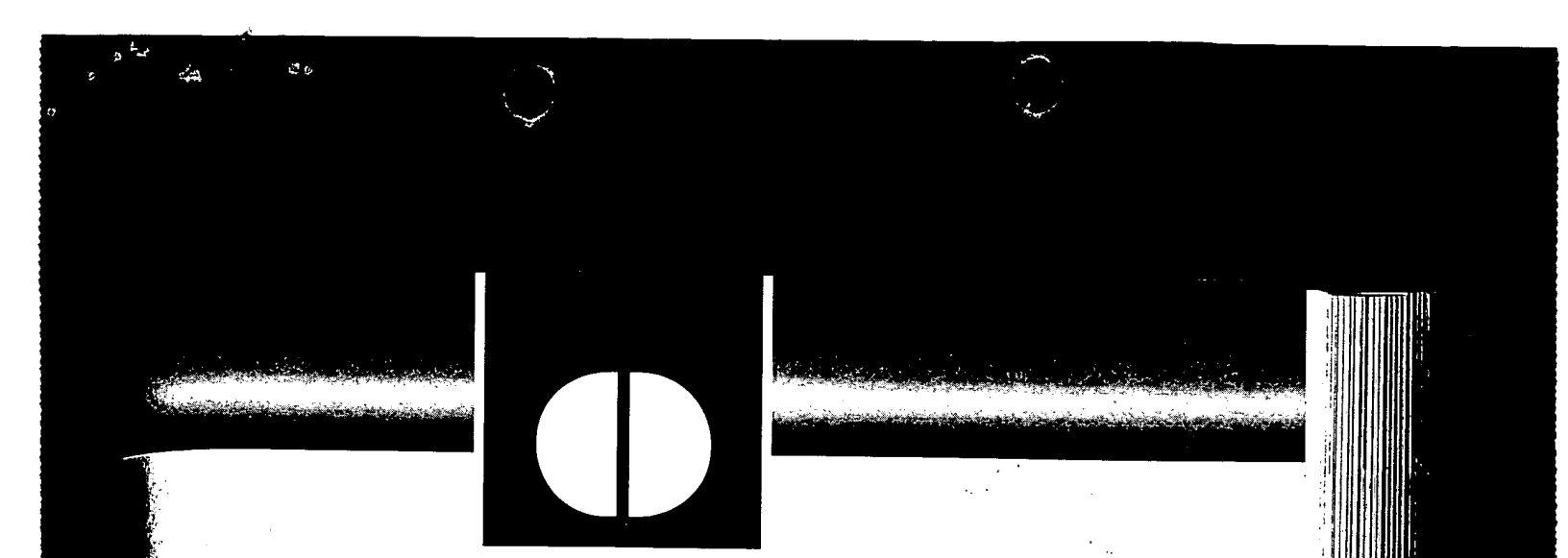
3 4 5 6 7 8 9 QMQM 2 1 0 9 8

Distributed to the book trade in Canada by Macmillan of Canada, a division of Canada Publishing Corporation.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office. Or contact Microsoft Press International directly at fax (425) 936-7329.

Acquisitions Editor: Kim Fryer
Project Editor: Maureen Williams Zimmerman, Anne Taussig
Technical Editors: Dail Magee Jr., Gary Nelson, Jean Ross, Jim Fuchs, John Conrow, Kurt Meyer, Robert Lyon, Roslyn Lutsch



ject \ob´jekt\ *n.* **1.** Short for object code achine-readable code). **2.** In object-oriented ogramming, a variable comprising both routines d data that is treated as a discrete entity. *See also* stract data type, module (definition 1), object-ented programming. **3.** In graphics, a distinct ity. For example, a bouncing ball might be an ject in a graphics program.

ject code \ob´jekt kōd`\ *n.* The code, gener-d by a compiler or an assembler, that was trans-d from the source code of a program. The term st commonly refers to machine code that can directly executed by the system's central pro-sing unit (CPU), but it can also be assembly lan-ge source code or a variation of machine code. *also* central processing unit.

t computer \ob´jekt kəm-pyo͞o`tər\ *n.* The puter that is the target of a specific communi-ns attempt.

t database \ob´jekt dā`tə-bās\ *n. See* object-ted database.

t Database Management Group \ob`jəkt -bās man´əj-mənt gro͞op`\ *n.* An organization promotes standards for object databases and s interfaces to object databases. *Acronym:* G (O`D-M-G´). *See also* Object Management p.

file \ob´jekt fīl`\ *n.* A file containing object usually the output of a compiler or an pler and the input for a linker. *See also* bler, compiler (definition 2), linker, object

ve-C \əb-jek`tiv-C´\ *n.* An object-oriented of the C language developed in 1984 by ox. It is most widely known for being the d development language for the NeXT g system. *See also* object-oriented pro-ing.

inking and embedding \ob`jekt lēnk`ēng bed´ēng\ *n. See* OLE.

Object Management Architecture \ob`jekt man`əj-mənt är´kə-tek-chur\ *n. See* OMA.

Object Management Group \ob`jekt maṇ´əj-mənt gro͞op`\ *n.* An international organization that endorses open standards for object-oriented applications. It also defines the Object Management Architecture (OMA), a standard object model for distributed environments. The Object Management Group was founded in 1989. *Acronym:* OMG (O`M-G´). *See also* object model (definition 3), OMA, open standard.

object model \ob´jekt mod`əl\ *n.* **1.** The structural foundation for an object-oriented language, such as C++. This foundation includes such principles as abstraction, concurrency, encapsulation, hierarchy, persistence, polymorphism, and typing. *See also* abstract data type, object (definition 2), object-oriented programming, polymorphism. **2.** The structural foundation for an object-oriented design. *See also* object-oriented design. **3.** The structural foundation for an object-oriented application.

object module \ob´jekt moj`o͞ol, mod`yo͞ol\ *n.* In programming, the object-code (compiled) version of a source-code file that is usually a collection of routines and is ready to be linked with other object modules. *See also* linker, module (definition 1), object code.

object-oriented \ob´jekt-ōr`ē-en-təd\ *adj.* Of, pertaining to, or being a system or language that supports the use of objects. *See also* object (definition 2).

object-oriented analysis \ob`jekt-ōr-ē-ent-əd ə-nal´ə-sis\ *n.* A procedure that identifies the component objects and system requirements of a system or process that involves computers and describes how they interact to perform specific tasks. The reuse of existing solutions is an objective of this sort of analysis. Object-oriented analysis generally precedes object-oriented design or object-oriented programming when a new object-oriented computer

system or new software is developed. *See also* object (definition 2), object-oriented design, object-oriented programming.

object-oriented database \ob`jekt-ōr-ē-ent-əd dā´tə-bās\ *n.* A flexible database that supports the use of abstract data types, objects, and classes and that can store a wide range of data, often including sound, video, and graphics, in addition to text and numbers. Some object-oriented databases allow data retrieval procedures and rules for processing data to be stored along with the data or in place of the data. This allows the data to be stored in areas other than in the physical database, which is often desirable when the data files are large, such as those for video files. *Acronym:* OODB (O`O-D-B´). *See also* abstract data type, class, object (definition 2). *Compare* relational database.

object-oriented design \ob`jekt-ōr-ē-en-təd də-zīn´\ *n.* A modular approach to creating a software product or computer system, in which the modules (objects) can be easily and affordably adapted to meet new needs. Object-oriented design generally comes after object-oriented analysis of the product or system and before any actual programming. *See also* object (definition 2), object-oriented analysis.

object-oriented graphics \ob`jekt-ōr-ē-en-təd graf´iks\ *n.* Computer graphics that are based on the use of graphics primitives, such as lines, curves, circles, and squares. Object-oriented graphics, used in applications such as computer-aided design and drawing and illustration programs, describe an image mathematically as a set of instructions for creating the objects in the image. This approach contrasts with the use of bitmapped graphics, in which a graphic is represented as a group of black-and-white or colored dots arranged in a certain pattern. Object-oriented graphics enable the user to manipulate objects as units. Because objects are described mathematically, object-oriented graphics can be layered, rotated, and magnified relatively easily. *Also called* structured graphics. *See also* graphics primitive. *Compare* bitmapped graphics, paint program.

object-oriented interface \ob`jekt-ōr-ē-en-təd in´tər-fās\ *n.* A user interface in which elements of the system are represented by visible screen entities, such as icons, that are used to manipulate the system elements. Object-oriented display in faces do not necessarily imply any relatio object-oriented programming. *See also* ob oriented graphics.

object-oriented operating system \ob`jekt-ō en-təd op´ər-ā-tēng si`-stəm\ *n.* An operating tem based on objects and designed in a way facilitates software development by third pa using an object-oriented design. *See also* ob (definition 2), object-oriented design.

object-oriented programming \ob`jekt-ōr- təd prō´gram`ēng\ *n.* A programming paradig which a program is viewed as a collection of crete objects that are self-contained collection data structures and routines that interact other objects. *Acronym:* OOP (ōōp, O`O-P). *also* C++, object (definition 2), Objective-C.

object-relational server \ob`jekt-rə-lā´shə- sər´vər\ *n.* A database server that supports ob oriented management of complex data types relational database. *See also* database server tional database.

object request broker \ob`jekt rə-kwest kər\ *n. See* ORB.

object wrapper \ob´jəkt rap`ər\ *n.* In ob oriented applications, a means of encapsul a set of services provided by a non-ob oriented application so that the encapsul services can be treated as an object. *See* object (definition 2).

oblique \ō-blēk´\ *adj.* Describing a style of created by slanting a roman font to simulate when a true italic font isn't available on the puter or printer. *See also* font, italic, roman.

OC3 \O`C-thrē´\ *n.* Short for **o**ptical **c**arrier 3 of several optical signal circuits used in the SO high-speed fiber-optic data transmission sys OC3 carries a signal of 155.52 Mbps, the mini transmission speed for which SONET and European standard, SDH, are fully interoper *See also* SONET.

OCR \O`C-R´\ *n. See* optical character recogni

octal \ok´təl\ *n.* The base-8 number system sisting of the digits 0 through 7, from the *octo,* meaning "eight." The octal system is us programming as a compact means of repres binary numbers. See Appendix E. *See also* (definition 2).